WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Monday, March 08, 2004

Hide?	Set Name	Query	Hit Count
		DB=USPT,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR	
☐	L44	L41.ab.	2
☐	L43	L41 and l15	2
☐	L42	L41.ab.	2
☐	L41	((buffer or fifo or queue) near3 (address$4 or locat$4) near5 (lookup adj (table or list)))	40
☐	L40	l18 same ((queue or buffer or fifo) near2 control$4)	1
☐	L39	l38 same (data near2 (transfer$4 or send$4 or transmit$4))	7
☐	L38	((queue or fifo or buffer) near2 control$4) same (lookup adj table)	56
☐	L37	l22 same (lookup near2 table)	0
☐	L36	(interrogat$4 near3 lookup near2 table)	6
☐	L35	(register$4 near2 (queue or fifo or buffer) near3 (lookup adj table))	8
☐	L34	L33 same ((queue or fifo or buffer) near2 control$4)	13
☐	L33	l31 same (data near2 (transfer$4 or send$4 or transmit$4))	30
☐	L32	l18 and L31	0
☐	L31	(register$4 near2 (queue or fifo or buffer) near3 table)	278
☐	L30	l23 and l15	23
☐	L29	l23.ab.	5
☐	L28	l6 and l22	0
☐	L27	l1 and l22	0
☐	L26	L23 and (lookup adj table)	7
☐	L25	L23 same l1	0
☐	L24	L23 same lookup	0
☐	L23	L22 same (register$4 near3 (queue or buffer or fifo))	161
☐	L22	((send or transmit) adj (queue or buffer or fifo)) same (receive adj (queue or buffer or fifo))	1739
☐	L21	(establish$4 near3 ((send or transmit) adj (queue or buffer or fifo)))	17
☐	L20	l18.clm.	5
☐	L19	l15 and L18	1
☐	L18	L17 same (data near2 (transfer$4 or send$4 or transmit$4))	41
☐	L17	(first near2 partition) same (second near2 partition)	4454
☐	L16	l2 and L15	8

☐	L15	l7 or l8 or l9 or l10 or l11 or l12 or l13 or L14	3572
☐	L14	714/11.ccls.	391
☐	L13	712/13.ccls.	110
☐	L12	709/236.ccls.	467
☐	L11	710/310.ccls.	287
☐	L10	710/36.ccls.	483
☐	L9	710/33.ccls.	409
☐	L8	710/52.ccls.	1160
☐	L7	710/5.ccls.	646
☐	L6	(logical near2 partition$4).ti.	127
☐	L5	l1.ti.	4
☐	L4	L3.ab.	13
☐	L3	L1 same (data adj processing adj system)	33
☐	L2	L1.ab.	46
☐	L1	((plurality or multiple) near3 (virtual or logical) near3 partitions)	192

END OF SEARCH HISTORY

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L40: Entry 1 of 1 File: USPT Sep 11, 2001

DOCUMENT-IDENTIFIER: US 6288794 B1
TITLE: Variable sample rate converter adaptable to rational ratios

Detailed Description Text (7):
FIG. 3 is a system diagram illustrating another embodiment of the present invention. In particular, FIG. 3 illustrates a sample rate converter 300 having various functional elements arranged in accordance with the present invention. A data control buffer 310 is used to control a flow of incoming data. In certain embodiments, data control buffer 310 is used to control the incoming plurality of data 210. In other embodiments, the data control buffer 310 is partitioned into at least two portions, a first partition 304 and a second partition 308. This partition may be viewed as that of a ping-pong style buffer in which data is transmitted into one of either the first partition 304 and the second partition 308, and data is transmitted from the other partition. Such an implementation serves to maximize the transfer of data into and out of the data control buffer 310.

First Hit Fwd Refs

Generate Collection Print

L39: Entry 5 of 7 File: USPT Mar 21, 2000

DOCUMENT-IDENTIFIER: US 6041397 A
TITLE: Efficient transmission buffer management system

Brief Summary Text (11):
Memory blocks have been divided into pairs of buffers holding related information for data transfer purposes. It is desirable to refer to a buffer pair using a single address to ensure a link between the buffers and for ease of processing. One traditional method of addressing a buffer pair is to use a lookup table, but this requires software intervention. Another method is to use a sequential access rule in which buffer N holds control information and buffer N+1 holds data, but this requires the two buffers to be the same size. Another method is to concatenate the control information and data in a single buffer, with the data starting at a known offset from the control information, but this requires an inefficient oversized buffer. Another problem with some of the above methods is that the addition of a single memory component requires reconfiguring the addressing scheme.

First Hit Fwd Refs

Generate Collection Print

L34: Entry 8 of 13 File: USPT Aug 8, 1978

DOCUMENT-IDENTIFIER: US 4106091 A
TITLE: Interrupt status indication logic for polled interrupt digital system

Detailed Description Text (18):
Table 1 defines the functions of the transmit data register 32, the receive data register 38, the control register 36, and the status register 34 for a presently preferred embodiment of the invention. The combinations of the RS and R/W inputs required to select each of the registers are indicated in the "Buffer Address" row of Table 1. The "Bus Line Number" designations refer to the conductors of bidirectional data bus 14 and the corresponding bits of the four abovementioned internal registers. Bits 0 and 1 of control register 36 are dedicated to selecting one of three divide clock ratios and to establishing a master reset function which reset all of the logic on the chip. Bits 2, 3 and 4 of control register 36 are dedicated to selecting one of eight different combinations of data word lengths, parity bits, and stop bits. Bits 5 and 6 of control register 36 control the "transmitter buffer empty" interrupt output, the state of the request-to-send (RTS) output and the transmission of a "Break" level (i.e., space). Bit 7 of the control register controls interrupts being caused by the "receiver data register full" indicator and by DCD. It should be noted that writing data into the transmit data register 32 causes the "transmit data empty" bit in status register 34 to go low and data can then be transmitted. Transfer of data therefrom causes the transmit data register empty bit to indicate empty. Upon receiving a complete character, data is automatically transferred to the empty receive data register 38 from receive shift register 56, which even causes the receive data register full bit in status register 34 to go high, allowing data to be read through bidirectional data bus 14. The nondestructive read cycle causes the "receive data register full" bit to be cleared. When the receive data register 38 is full, the automatic transfer of data from the receive shift register 56 is inhibited so that the contents of receive data register 38 remains valid.

L35: Entry 6 of 8 File: USPT Oct 4, 1994

DOCUMENT-IDENTIFIER: US 5353402 A
TITLE: Computer graphics display system having combined bus and priority reading of video memory

CLAIMS:

11. A graphics display system as defined in claim 10, further including a lookup table for receiving data from the FIFO register, for using the received data from the FIFO to look up pixel data therein, a digital-to-analog converter for receiving said pixel data and for providing analog signals to a display for display thereon.

First Hit Fwd Refs

Generate Collection Print

L29: Entry 1 of 5 File: USPT Jul 25, 2000

DOCUMENT-IDENTIFIER: US 6094696 A
TITLE: Virtual serial data transfer mechanism

Abstract Text (1):
A plurality of data devices are interfaced to a microprocessor using a serial data transfer mechanism. The parallel data from the data devices is serialized. The serial data streams are multiplexed via a data multiplexer. An index signal identifies the data device from which the serial data is received/transmit. When a receive buffer is at a predefined level of emptiness, a bit associated with that buffer is asserted. Likewise, when a transmit buffer is at a predefined level of emptiness, a bit within the index register associated with the transmit buffer is asserted. The assertion of a bit within the index register generates an interrupt. A CPU core receives the interrupt signal and reads the index register to determine which buffers need servicing. The CPU core deasserts one bit of the index register, which indicates the CPU core is going to service the buffer associated with that bit. If the bit in the index register is associated with a receive buffer, the deassertion of the bit causes the receive buffer to output receive data to the CPU core. Likewise, if the bit deasserted by the CPU core is associated with the transmit buffer, the deassertion of the bit within the index register causes the transmit buffer to input data from the CPU core.

First Hit Fwd Refs

Generate Collection Print

L34: Entry 3 of 13 File: USPT Mar 11, 1997

DOCUMENT-IDENTIFIER: US 5610808 A
**** See image for Certificate of Correction ****
TITLE: Hard disk drive controller employing a plurality of microprocessors

Detailed Description Text (39):
Data transfers by the host microcontroller 52 are passed via the host microcontroller interface unit 202 through a selectable, bi-directional one or two byte depth FIFO unit 224 via data bus 226 to the data multiplexer 222 via bus 228. Table III lists the buffer control registers accessible to the host microcontroller 52 in the buffer register unit 232. Since, in the preferred embodiments of the present invention, the host and low-level microcontrollers are both based on 8-bit processors, the data FIFO 224 appears as two byte-wide registers locations in the memory map of the host processor 52.

Detailed Description Text (48):
In a manner similar to the host microcontroller interface unit 202, the low-level microcontroller interface unit 206 exchanges control information with the arbitration and buffer control unit 200 via control lines 208, transfers data via bus 242 and a location address via the partial address bus 234. Buffer memory 68 access requests are preferably treated at a priority level below that of the host microcontroller interface unit 202. The data bus 242 connects through another bi-directional, one or two byte deep FIFO 244 that is controlled by the low-level microcontroller interface unit 206 via control line 246. The FIFO 244 connects via an internal data bus 248 to the buffer register 232 and a bi-directional data port of the data multiplexer 222. The data bus 248 is further connected to an access port of the buffer register 232 and, separately, to an access port of a sequencer control and status register 250. Finally, through the low-level microcontroller interface unit 206, the microcontroller 50 has access to the interprocessor register set shown in Tables I and II and to the buffer access register set shown in Table IV.

First Hit Fwd Refs

Generate Collection Print

L34: Entry 4 of 13 File: USPT May 2, 1995

DOCUMENT-IDENTIFIER: US 5412666 A
**** See image for Certificate of Correction ****
TITLE: Disk drive data path integrity control architecture

Detailed Description Text (39):
Data transfers by the host micrcontroller 52 are passed via the host microcontroller interface unit 202 through a selectable, bi-directional one or two byte depth FIFO unit 224 via data bus 226 to the data multiplexer 222 via bus 228. Table III lists the buffer control registers accessible to the host microcontroller 52 in the buffer register unit 232. Since, in the preferred embodiments of the present invention, the host and low-level microcontrollers are both based on 8-bit processors, the data FIFO 224 appears as two byte-wide registers locations in the memory map of the host processor 52.

Detailed Description Text (48):
In a manner similar to the host microcontroller interface unit 202, the low-level microcontroller interface unit 206 exchanges control information with the arbitration and buffer control unit 200 via control lines 208, transfers data via bus 242 and a location address via the partial address bus 234. Buffer memory 68 access requests are preferably treated at a priority level below that of the host microcontroller interface unit 202. The data bus 242 connects through another bi-directional, one or two byte deep FIFO 244 that is controlled by the low-level microcontroller interface unit 206 via control line 246. The FIFO 244 connects via an internal data bus 248 to the buffer register 232 and a bi-directional data port of the data multiplexer 222. The data bus 248 is further connected to an access port of the buffer register 232 and, separately, to an access port of a sequencer control and status register 250. Finally, through the low-level microcontroller interface unit 206, the microcontroller 50 has access to the interprocessor register set shown in Tables I and II and to the buffer access register set shown in Table IV.

First Hit Fwd Refs
End of Result Set

Generate Collection Print

L19: Entry 1 of 1 File: USPT Feb 10, 1998

DOCUMENT-IDENTIFIER: US 5717942 A
TITLE: Reset for independent partitions within a computer system

Detailed Description Text (46):
A storage controller reset may only clear the input logic and the output logic associated with that storage controller. For example, a reset from storage controller-0 on interface 204 may only reset input logic 200 and output logic 230. This allows the XBAR logic associated with one partition to be cleared without affecting data transfers occurring in the other partitions. For example, the XBAR logic that is associated with the first partition 83 containing storage controller-0 and storage controller-1 may be cleared via the reset signals on interfaces 204 and 210. At the same time, transfers involving units in the second partition 85 containing storage controller-2 and storage controller-3 can continue without interruption.

Detailed Description Text (59):
A reset may be provided by storage controller-0 via interface 204. The reset may be provided to control and destination validation logic block 280, error logic 300, and request register 304. As stated above, a storage controller may reset only the input logic and the output logic within the XBAR interface block 86 which is associated with that storage controller. For example, a reset from storage controller-0 on interface 204 may only reset input logic 200 and output logic 230. This allows the XBAR logic associated with one partition to be cleared without affecting data transfers occurring in another partition. That is, the XBAR logic associated with the first partition 83 containing storage controller-0 and storage controller-1 may be cleared via the reset signals on interfaces 204 and 210 (see FIG. 5). At the same time, transfers involving units in the second partition 85 containing storage controller-2 and storage controller-3 may continue without interruption.

Detailed Description Text (67):
A reset may be provided by storage controller-0 via interface 204. The reset may be provided to request stack 350 and priority/control logic block 358. As stated above, a storage controller may reset only the input logic and the output logic within the XBAR interface block 86 which is associated with that storage controller. For example, a reset from storage controller-0 on interface 204 may only reset input logic 200 and output logic 230. This allows the XBAR logic associated with one partition to be cleared without affecting data transfers occurring in another partition. That is, the XBAR logic associated with the first partition 83 containing storage controller-0 and storage controller-1 may be cleared via the reset signals on interfaces 204 and 210 (see FIG. 5). At the same time, transfers involving units in the second partition 85 containing storage controller-2 and storage controller-3 may continue without interruption.

Current US Original Classification (1):
712/13

First Hit Fwd Refs

Generate Collection Print

L20: Entry 1 of 5 File: USPT Dec 23, 2003

DOCUMENT-IDENTIFIER: US 6667973 B1
TITLE: Flexible SONET access and transmission systems

CLAIMS:

6. The SONET network interface of claim 1, wherein the bus is a common bus and the SONET interface further comprises: a first partition bus for interfacing the at least one HSU to a predetermined number of the at least two LSUs, the first partition bus being partitioned into a first bus for interfacing the at least one HSU to a first subset of the at least two LSUs and a second bus for interfacing the at least one HSU to a second subset of the at least two LSUs; and a second partition bus for interfacing the at least one HSU to a predetermined number of the at least two LSU units, the second partition bus being partitioned into a third bus for interfacing the at least one HSU unit to the first subset of the at least two LSUs and a fourth bus for interfacing the at least one HSU unit with the second subset of the at least two LSUs; wherein the transmit data is transmitted between the LSUs via the first and second or third and forth buses through the at least one HSU acting as a physical bridge therebetween.

14. The method of claim 9, wherein the bus is a common bus and the method further comprises the steps of: interfacing the at least one HSU to a predetermined number of the at least two LSUs via a first partition bus, the first partition bus being partitioned into a first bus for interfacing the at least one HSU to a first subset of the at least two LSUs and a second bus for interfacing the at least one HSU to a second subset of the at least two LSUs; and interfacing the at least one HSU to a predetermined number of the at least two LSU units via a second partition bus, the second partition bus being partitioned into a third bus for interfacing the at least one HSU unit to the first subset of the at least two LSUs and a fourth bus for interfacing the at least one HSU unit with the second subset of the at least two LSUs; wherein the transmit data is transmitted between the LSUs via the first and second or third and forth buses through the at least one HSU acting as a physical bridge therebetween.

First Hit Fwd Refs

Generate Collection Print

L20: Entry 2 of 5 File: USPT Sep 30, 2003

DOCUMENT-IDENTIFIER: US 6628649 B1
TITLE: Apparatus and methods providing redundant routing in a switched network device

CLAIMS:

4. The data communications device of claim 3 wherein: the first route provided within the first partition is established as a corresponding first switched connection; the second route provided within the second partition is established as a corresponding second switched connection; and wherein the first and second switched connections handle data transfers concurrently and according to a multipath routing protocol that distributes data between the first and second routes, such that the first and second switch control mechanisms support routes for distinct parallel networks that operate to transfer data concurrently through the data communications device.

5. The data communications device of claim 4, further including: means for detecting an inability of the first switch control mechanism to support the first route within the first partition; means, in response for the means for detecting, for re-routing a data stream transferred using the first route within the first partition to the second route within the second partition.

6. The data communications device of claim 5, further including: means, in response to the means for detecting, for de-allocating data switch resources from the first partition used to support the first connection; and means, in response for the means for de-allocating, for allocating at least a portion of the data switch resources de-allocated from the first partition to the second partition, such that the portion of data switch resources that are de-allocated from the first partition are allocated to the second partition in order to support data transfers on the second route provided by the second connection.

7. The data communications device of claim 2 wherein: the first route provided within the first partition is established as a corresponding first switched connection; the second route provided within the second partition has no corresponding second switched connection in the second partition; and wherein the first switched connection handles data transfers according to a unipath routing protocol that initially selects the first route to perform data transfers, such that the first and second switch control mechanisms each support routes for distinct parallel networks, but that only one of such routes is selected according to the unipath routing protocol to initially transfer data through the data communications device.

8. The data communications device of claim 7, further comprising: means for detecting an inability of the first switch control mechanism to support the first route within the first partition; and means, in response to the means for detecting, to establish a second connection in the second partition according to the second route supported by the second switch control mechanism, the second connection used to provide data transfers of a stream of data formerly transferred through the first connection.

9. The data communications device of claim 8, further comprising: means for de-allocating data switch resources from the first partition used to support the first connection, and means for removing the first connection from the first partition; and means, in response for the means for de-allocating, for allocating the data switch resources de-allocated from the first partition to the second partition, such that the data switch resources that are de-allocated from the first partition are allocated to the second partition in order to support data transfers on the second route provided by the second connection.

First Hit Fwd Refs

Generate Collection Print

L20: Entry 3 of 5 File: USPT Oct 8, 2002

DOCUMENT-IDENTIFIER: US 6463584 B1
TITLE: State copying method for software update

CLAIMS:

1. A software processing device including update functionality, comprising: memory means subdivided into an executing memory partition means for storing a first group of software modules and related data, and a standby memory partition means for storing a second group of software modules and related data; update control means adapted to update a state of new software in the standby memory partition means to a state of old software in the executing memory partition means during continuation of execution of the old software; transfer means for scalably transferring data from the executing memory partition means to the standby memory partition means; and switching means and state comparison means to instantly switch to execution of new software gas soon as a same state is detected for the standby memory partition means and the executing memory partition means by the state comparison means.

3. A software processing device including update functionality, comprising: memory means subdivided into an executing memory partition means for storing a first group of software modules and related data, and a standby memory partition means for storing a second group of software modules and related data; update control means adapted to update a state of new software in the standby memory partition means to a state of old software in the executing memory partition means during continuation of execution of the old software; and transfer means for scalably transferring data from the executing memory partition means to the standby memory partition means, wherein each memory partition means is assigned at least one take over means to carry out default actions in case data related to old software is only partly transferred such that the take over means is activated immediately after switch over.

4. A software processing device including update functionality, comprising: memory-means subdivided into an executing memory partition means for storing a first group of software modules and related data, and a standby memory partition means for storing a second group of software modules and related data; update control means adapted to update a state of new software in the standby memory partition means to a state of old software in the executing memory partition means during continuation of execution of the old software; and transfer means for scalably transferring data from the executing memory partition means to the standby memory partition means, wherein the transfer means copies data either unchanged or after conversion into a new representation for the new software.

7. A software processing device including update functionality, comprising: memory means subdivided into an executing memory partition means for storing a first group of software modules and related data, and a standby memory partition means for storing a second group of software modules and related data; update control means adapted to update a state of new software in the standby memory partition means to a state of old software in the executing memory partition means during continuation of execution of the old software; transfer means for scalably transferring data from the executing memory partition means to the standby memory partition means; and switching means for switching between software on the executing memory

partition means and the standby memory partition means, wherein the update control means repeatedly updates until the switching means switches to execution of the new software to keep track of a changing state in the executing memory partition means.

28. A state copying method for a distributed computation environment comprising one main processor means and at least one remote processor means, comprising the steps of: updating new software into a first memory partition means of the remote processor means; updating a state of the new software to achieve a match with a state of the main processor means while continuing execution of software in the main processor means, wherein data is scalably transferred between the first memory partition means and a second memory partition means; and switching the execution of software in the remote processor means to the new software as soon as a match with the state of the main processor means is achieved.

First Hit Fwd Refs

Generate Collection Print

L20: Entry 4 of 5 File: USPT Oct 17, 1989

DOCUMENT-IDENTIFIER: US 4875155 A
TITLE: Peripheral subsystem having read/write cache with record access

CLAIMS:

2. A method for improving data transfer between host and a peripheral subsystem including a relatively fast access cache and a relatively slow access backing store having a plurality of addressable data storage devices, each of the devices having a plurality of addressable data storage tracks, comprising the steps of: defining a first table having at least one bit for each device in said back store; defining a second table having at least one bit for each track of each said device; defining a third table having a plurality of entries, each entry having indicia of location of a record in said cache; defining a fourth table having a first and second partitions, said first partition being an array having N bytes, N being equal to a total of record numbers storable in the backing store, and said second partition comprising a plurality of entries, each said entry comprising one or more bytes of data containing record identifying indicia, there being at least a number of said entries in said second partition, equal to a maximum number of records to be stored on a track of a device in said back store; defining an area of said cache as a track buffer, said buffer having sufficient storage capacity to store all records from one track of a device in the back store; determining for each record transfer between said host and said subsystem by reference to said first, second and fourth tables whether there is a version of said record in cache including scanning said last-mentioned tables; and checking said third table for ascertaining whether or not said record in said back store for determining if such version does or does not exist in the cache.

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L20: Entry 5 of 5 File: USPT Mar 26, 1985

DOCUMENT-IDENTIFIER: US 4507781 A
TITLE: Time domain multiple access broadcasting, multipoint, and conferencing communication apparatus and method

CLAIMS:

1. An apparatus for selectively carrying out point-to-point or broadcasting connections in a TDMA satellite communications network between remote earth stations, comprising:

a transmit burst buffer having a plurality of data partitions, each corresponding to a particular local transmitting port, with a first port selectively transmitting point-to-point and a second port selectively transmitting in a broadcast mode to destination ports at said second earth stations, and said transmit burst buffer at said first station having an output connected to TDMA burst transmission means;

a transmit space signaling buffer at said first earth station having a first partition storing a direct destination address corresponding to the data to be transmitted point-to-point from said first local port to a first port in said second earth station and a second partition in said transmit space signaling buffer for storing an indirect broadcast address corresponding to the data stored in said second partition in said transmit burst buffer from said second local port for a broadcast connection, said transmit space signaling buffer having an output connected to said TDMA burst transmission means;

burst logic means at said first station having an input connected to said transmit burst buffer and said transmit space signaling buffer for accessing said point-to-point data in said first partition of said transmit burst buffer and appending to it said direct destination address from said first partition in said transmit space signaling buffer and for accessing said broadcast data stored in said second partition of said transmit burst buffer and appending to it said indirect broadcast address stored in said second partition of said transmit space signaling buffer and directing said data and addresses to said TDMA burst transmission means;

a comparison means in said second earth station having an input connected to a TDMA burst receiving means, for comparing the first direct destination address from said first earth station with the destination address of said second earth station and routing said first point-to-point data received from said first earth station to a first partition in a receive burst buffer in said second earth station;

said comparison means comparing said second indirect broadcast address received from said first earth station with a stored broadcast address and routing said second broadcast data received from said first earth station to a second, common location in said receive burst buffer in response to indirectly addressing a broadcast memory in said second earth station having an input connected to said comparison means and an output connected to said receive burst buffer for accessing said common location in said receive burst buffer;